(A): Amendments to the specification:

Amend the second paragraph on page 1 to read as follows:

Related Applications:

This invention is related to an application entitled: Multiprocessor System with Dynamic Cache Coherency Regions, USSN 10/603,251, filed June 25, 2003, filed concurrently herewith and referenced herein.

Amend the paragraph on Page 1, containing line 18 to read as follows:

The description set forth in ~~these~~ this related co-pending application is hereby incorporated into the present application by this reference.

Amend page 2, penultimate paragraph containing line 32 to read as follows:

Although caches have worked well for multiprocessor systems with a moderate number of processors, prior-art multiprocessor designs do not scale well when extended to large numbers of processors for many important workloads including the transaction and database workload simulated by ~~the~~ a TPC-C benchmark.

Amend page 8, after line 19, by insertion of the following paragraphs:

Figure 8 illustrates particularly how supervisor software moves a software process out of one coherency region that is no longer going to be used by that software process and into another coherency region that has been created to cover the same address space as the first but which will include a new set of processing nodes.

Figure 9 illustrates particularly incoming storage requests, some of which misses all of the cashes in an originator's coherency region and some of which hit in an originator's coherency region.

Figure 10 illustrates particularly processing nodes which are able to identify incoming storage requests which are target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node.

Amend page 6 by replacing the first full paragraph with the following:

In fulfilling these determined needs, we have hardware coherency controls which enable a system which uses multiple cache coherency regions to operate without the use of cache purges during some operations which move software processes between coherency regions. The current invention works for the case (Figure 8) where a software process is moved out of one coherency region that is no longer going to be used and into another that has been created to cover the same address space as the first but which will include a new set of processing nodes. The preferred embodiment of our invention works to allow a supervisor program to move a software process from one coherency region encompassing

one set of processing nodes to another coherency region encompassing another set of processing nodes without requiring cache purges of caches in any of the processing nodes. If the destination coherency region contains fewer hardware processing nodes than the original coherency region, the size of the coherency region has been effectively been reduced.

Amend page 7 by replacing the first full and second paragraphs with the following:

Processing nodes which use the invention, as illustrated particularly with Figures 9 and 10, are able to identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node. In the preferred embodiment this information allows a processing node to identify cache lines that are no longer actively used by any software processes on that node and to change the cache entries to invalid in response to a storage request from outside the coherency region.

The advantages of the invention, as illustrated particularly with Figures 9 and 10, are numerous. One advantage of the invention is that it eliminates the need for cache control hardware that would otherwise be required to perform selective purges of caches when moving software processes between cache coherency regions. A second advantage is that the invention allows all of the caches in a system to continue processing coherency transactions while the coherency boundaries for a software process are effectively changed. A third advantage is

that cache lines belonging to a software process that is no longer actively being dispatched on a given node can be identified and invalidated thereby enabling their reuse.

Amend page 12 beginning at line 4 to read as follows:

Alternative approaches may be found in the related patent application. The referenced related patent application USSN 10/603,251 uses a combination of software control programs and hardware mode bits to define dynamic coherency boundaries in a computing system which utilizes more than one processing node. The coherency boundaries can be adjusted to create coherency regions that cover any number of processing nodes, from one node to all the nodes in the entire system. The related application also describes how multiple coherency regions can be defined, each operating on a private address space. The coherency regions can be expanded to include additional nodes at any time during system operation. The coherency regions can also be reduced in size by removing a processing node from that region, while following the required procedures. Included in those procedures is the need to purge some cache entries in the processing node which is about to be removed from the coherency region. The cache entries which need to be removed are only those that hold cached copies of main storage lines that are part of the coherency region that is being reduced in size. Caches which are unable to perform a selective purge based upon the identification of the coherency region that “owns” the cached line must be purged completely. The selective purges require additional cache controller hardware as compared to prior-art designs. The preferred embodiment illustrated here is applicable to a

processing system taking advantage of the ability to eliminate the need for the cache purges when moving a software process between two distinct sets of processing nodes.

The related patent application USSN 10/603,251 describes how the supervisor software must change the "cache coherence mode" of the processor when dispatching a software process that uses a cache coherency

Replace the paragraph beginning as the last partial paragraph on page 13 and continuing to be the first partial paragraph on page 14 with the following paragraph:

The current invention alters the logic which is used to decide which processing nodes must examine any particular storage request, as compared to the related patent application. In the referenced related patent application the coherency region, as expressed by the mode bits, ~~were~~ was used to make the determination. In the current preferred embodiment of the invention (See Figure 9 and 10) this is changed so that any storage request which misses all of the caches in the originator's coherency region is then sent on to all processing nodes in the entire system, regardless of the setting of the mode bits. Requests which hit in the originator's coherency region but which do not have the correct cache state do not need to be sent outside the coherency region. An example of this latter case is a storage request which intends to alter a line of storage but which finds during the course of a storage

transaction that a cache within its coherency region has a copy of the line that is marked as shared. The cache state transitions of the current invention are set up to ensure that a cache line cannot be marked as shared in two separate coherency regions. When the supervisor program is moving a coherency region from one set of processing nodes to another set of processing nodes it is effectively leaving behind cache entries for the coherency region on the old nodes. The current invention works to ensure that these old cache entries will be seen by requests originating from the new processing nodes and that cache entries for the same main storage addresses will not be established in the new processing nodes until the old entries are invalidated.

Amend the specification by inserting the following paragraphs after the first full paragraph of page 15 as the last paragraphs of the detailed description:

Figure 8 illustrates how supervisor software moves a software process out of one coherency region that is no longer going to be used by that software process and into another coherency region that has been created to cover the same address space as the first but which includes a new set of processors or nodes. As illustrated in Figure 8, at step 810 the process stops dispatch of process 0 on coherency region A (22 in Figure 2). During step 820 the supervisor software changes the Coherency Mode Bits for process 0 and alters the Active Coherency Region Tables (ACRTs)

on the node controllers of the system. The Coherency Region ID of process 0 is removed from the ACRTs of Node 2 and Node 3 (Figure 2). The Coherency Region ID of process 0 is added to the ACRT of Node 0. Thus the changes made in 820 allows the beginning of dispatch at step 830 of the process 0 on a new set of processors on coherency region B (18 in Figure 1).

Now, in response to the ACRT changes, processing nodes 2 and 3 are able to identify incoming storage requests which target lines that are no longer part of the address space of any software process that is currently enabled by the supervisor software to be dispatched on the node. This allows the caches attached to the nodes to identify cache lines that are no longer actively used by any software processes on that processing node and to change the cache entries for that processing node to invalid in response to a storage request from outside the coherency region. This method also allows all of the caches in said multiprocessor computer system to continue processing coherency transactions while the coherency boundaries for a software process are effectively changed. The method also identifies and invalidates cache lines belonging to a software process that is no longer actively being dispatched on a given processing node thereby enabling their reuse.

Figure 9 shows how this is achieved. Coherency requests are first broadcast only within the coherency region specified by the Coherency Mode Bit setting of the processor which originates the storage request (910). The originating processor collects the initial responses from the other processors in the coherence region and determines whether the request has missed all of the caches in the coherence region (920). Any storage request which misses all of the caches in an originating processor's coherency

region is rebroadcast (930) to all processing nodes in the entire system, regardless of the setting of the coherency mode bits. The Coherency Region ID of the process which initiated the request is sent along with the storage request during this second broadcast. If it is determined that the first storage request hit in one of the caches of the initial coherency region the request is completed using just those responses (940) with no need to broadcast the request outside of the coherency region. Thus, as shown in Figure 10 when a node receives an incoming storage request (1010) that includes a Coherency Region ID, it tests (1020) to determine if the Coherency Region ID matches an entry in the node's Active Coherency Region Table. If it does then the node provides a normal response to the coherency request (1040). If the Coherency Region ID does not match any entry in the node's Active Coherency Region Table then the node has determined that any cache entry associated with the request must be "left over" from a process that has been moved to a new coherency region. In this case (1030), the node responds to the coherency request and then invalidates the local copy and/or performs a castout of dirty lines. The actions described in Figure 9 will cause any line that was "left over" in the "old" coherency region, the coherency region of the process before the process was moved, to be removed from the "old" coherency region after the first attempted access to the same line by the process in the new coherency region that hosts the process after it has been moved.

Amend the ABSTRACT OF THE DISCLOSURE AS FOLLOWS:

ABSTRACT OF THE DISCLOSURE: